PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification [6] **:** **G06F 17/27**	**A1**	**(11) International Publication Number:** **WO 99/01828** **(43) International Publication Date:** 14 January 1999 (14.01.99)

(21) International Application Number: PCT/EP98/04153

(22) International Filing Date: 6 July 1998 (06.07.98)

(30) Priority Data:
9714126.1 4 July 1997 (04.07.97) GB
9800791.7 14 January 1998 (14.01.98) GB

(71) Applicant *(for all designated States except US)*: XEROX CORPORATION [US/US]; Xerox Square, Rochester, NY 14544 (US).

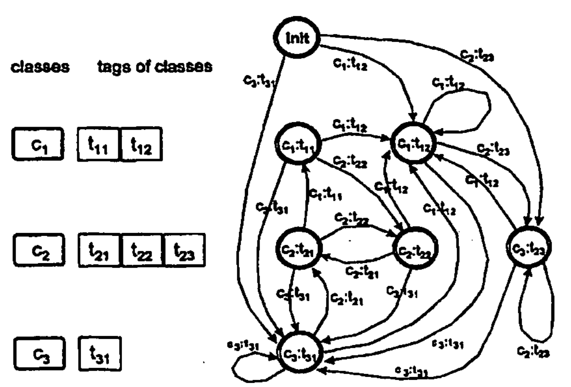
(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: KEMPE, André [DE/FR]; 259, chemin de Viers, F-38330 Biviers (FR).

(74) Agent: GRÜNECKER, KINKELDEY, STOCKMAIR & SCHWANHÄUSSER; Maximilianstrasse 58, D-80538 München (DE).

(81) Designated States: JP, US, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: FSTs APPROXIMATING HIDDEN MARKOV MODELS AND TEXT TAGGING USING SAME



(57) Abstract

A sequential transducer, derived from a Hidden Markov Model, that closely approximates the behavior of the stochastic model. The invention provides (a) a method (called n–type approximation) of deriving a simple finite–state transducer which is applicable in all cases, from HMM probability matrices, (b) a method (called s–type approximation) for building a precise HMM transducer for selected cases which are taken from a training corpus, (c) a method for completing the precise (s–type) transducer with sequences from the simple (n–type) transducer, which makes the precise transducer applicable in all cases, and (d) a method (called b–type approximation) for building an HMM transducer with variable precision which is applicable in all cases. This transformation is especially adavantageous for part–of–speech tagging because the resulting transducer can be composed with other transducers that encode correction rules for the most frequent tagging errors. The speed of tagging is also improved. The described methods have been implemented and successfully tested on six languages.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

FSTs approximating Hidden Markov Models and text tagging using same

The present invention relates to computer-based text processing, and more particularly to techniques for part-of-speech tagging by finite state transducer (FST) derived from a Hidden Markov Model (HMM).

Part-of-speech tagging of machine-readable text, whereby tags are applied to words in a sentence so as to identify a word's part-of-speech (e.g. noun-singular, verb-present-1st person plural), are known. Such tags are typically of a standardised form, such as specified under the Text Encoding Initiative (TEI). A text or corpus, once tagged, finds use, for example, in information retrieval from the text and statistical analysis of the text.

Probabilistic HMM based taggers are known from Bahl and Mercer (1976) and Church (1988); see Section H: References at the end of this disclosure. They select among the part-of-speech tags that are found for a word in the dictionary, the most probable tag based on the word's context, i.e. based on the adjacent words.

However, a problem with a conventional HMM based tagger is that it cannot be integrated with other finite state tools which perform other steps of text analysis/manipulation.

One method is to generate an FST performing part-of-speech tagging from a set of rules written by linguists (Chanod and Tapanainen, 1995). This, however, takes considerable time and produces very large automata (or sets of automata that cannot be integrated with each other for reasons of size) which perform the tagging relatively slowly. Such an FST may also be non-deterministic, i.e. for an input sentence it may provide more than one solution or no solutions at all.

The present invention provides methods to approximate a Hidden Markov Model (HMM) used for part-of-speech tagging, by a finite-state transducer. (An identical model of an HMM by a transducer (without weights) is in many cases impossible.) In specific embodiments there are provided (a) a method (called n-type approximation) of deriving a simple finite-state transducer which is applicable in all cases, from HMM probability matrices, (b) a method (called s-type approximation) for building a precise HMM transducer for selected cases which are taken from a training corpus, (c) a method for completing the precise (s-type) transducer with sequences from the simple (n-type) transducer, which makes the precise transducer applicable in all cases, and (d) a method (called b-type approximation) for building an HMM transducer with variable precision which is applicable in all cases.

The invention provides a method of generating a text tagging FST according to any of claims 1, 5 and 9 of the appended claims, or according to any of the particular embodiments described herein.

The invention further provides a method of generating a composite finite state transducer by composing the HMM-derived FST with one or more further text-manipulating FSTs.

The invention further provides a method of tagging a text using the aforementioned HMM-derived FST or the composite FST.

The invention further provides a text processing system according to claim 11 of the appended claims, and a recordable medium according to claim 12 of the appended claims.

The HMM transducer builds on the data (probability matrices) of the underlying HMM. The accuracy of the data collected in the HMM training process has an impact on the tagging accuracy of both the HMM itself and the derived transducer. The training of this HMM can be done on either a tagged or untagged corpus, and is not discussed in detail herein since it is exhaustively described in the literature (Bahl and Mercer, 1976; Church, 1988).

An advantage of Finite-State taggers according to the invention is that tagging speed when using transducers is up to five times higher than when using the underlying HMM (cf. section F hereinbelow).

However, a main advantage of the invention is that integration with tools performing further steps of text analysis is possible: transforming an HMM into a FST means that this transducer can be handled by the finite state calculus and therefore directly integrated into other finite-state text processing tools, such as those available from XEROX Corp., and elsewhere. Since the tagger is in the middle of a chain of text analysis tools where all the other tools may be finite-state-based (which is the case with text processing tools available from XEROX Corp.), converting the HMM into an FST makes this chain homogeneous, and thus enables merging the chain's components into one single FST, by means of composition.

In particular, it is possible to compose the HMM-derived transducer with, among others, one or more of the following transducers that encode:

- correction rules for the most frequent tagging errors in order to significantly improve tagging accuracy. These rules can either be extracted automatically from a corpus (Brill, 1992) or written manually (Chanod and Tapanainen, 1995). The rules may include long-distance dependencies which are usually not handled by HMM taggers.
- further steps of text analysis, e.g. light parsing or extraction of noun phrases or other phrases (Ait-Mokhtar and Chanod, 1997).
- criteria which decide on the relevance of a corpus with respect to a particular query in information retrieval (e.g. occurrence of particular words in particular syntactic structures).

These transducers can be composed separately or all at once (Kaplan and Kay, 1994). Such composition enables complex text analysis to be performed by a single transducer: see EP-A-583,083.

It will be appreciated that the algorithms described herein may find uses beyond those discussed with respect to the particular embodiments discussed below: i.e. on any kind of analysis of written or spoken language based on both finite-state technology and HMMs, such as corpus analysis, speech recognition, etc. The algorithms have been fully implemented.

Embodiments of the invention will now be described, by way of example, with reference to the accompanying drawings, in which:

Figure 1 illustrates decisions on tags with an n-type transducer, according to one embodiment of the invention;

Figure 2 illustrates schematically the generation of an n-type transducer in accordance with an embodiment of the invention;

Figure 3 illustrates the procedure of building an n-type transducer in accordance with an embodiment of the invention;

Figure 4 is a diagram illustrating class subsequences of a sentence;

Figure 5 is a schematic diagram of the steps in the procedure of building an s-type transducer, in accordance with an alternative embodiment of the invention;

Figure 6 is a diagram illustrating the disambiguation of classes between two selected tags;

Figure 7 is a diagram illustrating valid paths through the tag space of a sentence.

Figure 8 is a diagram illustrating b-type sequences.

Figure 9 is a schematic flow chart illustrating the procedure of building a b-type transducer;

Figure 10 is an illustration of the tagging of a sentence using either the n-type transducer formed in Fig. 3 or the s-type transducer formed in Fig. 5 or the b-type transducer formed in Fig. 9; and

Figure 11 is a schematic flow chart of the steps involved in the procedure, in accordance with an embodiment of the invention, of tagging a text corpus with a finite state tagger using an HMM transducer.

A. System configuration

It will be appreciated that the techniques according to the invention may be employed using conventional computer technology. It will be appreciated that the invention may be implemented using a PC running Windows™, a Mac running MacOS, or a minicomputer running UNIX, which are well known in the art. For example, the PC hardware configuration is discussed in detail in *The Art of Electronics*, 2nd Edn, Ch. 10, P. Horowitz and W. Hill, Cambridge University Press, 1989. The invention has been implemented in C on a Sun Sparc 20 workstation running UNIX.

B. FST derivation

The invention will be described in by reference to three methods of deriving a transducer for part-of-speech tagging from an HMM. These methods and transducers are referred to herein as *n-type, s-type and b-type.*

An HMM used for tagging encodes, like a transducer, a relation between two languages. One language contains sequences of ambiguity classes obtained by looking up in a lexicon all the words of a sentence. The other language contains sequences of tags obtained by statistically disambiguating the class sequences. From outside, an HMM tagger behaves like a sequential transducer that deterministically maps every sequence of ambiguity classes (corresponding to a sentence) into a unique sequence of tags, e.g.:

[DET] [ADJ,NOUN] [ADJ,NOUN] ...... [END]

DET ADJ NOUN ...... END

C. n-Type Transducers

This section presents a method that approximates a first order Hidden Markov Model (HMM) by a finite state transducer (FST), referred to as an n-type approximation. Figure 1 illustrates decisions on tags with an n-type transducer, according to one embodiment of the invention.

As in a first order HMM we take into account initial probabilities π, transition probabilities a and class (i.e. observation symbol) probabilities b. However, probabilities over paths are not estimated. Unlike in an HMM, once a decision is made, it influences the following decisions but is itself irreversible.

Figure 1 illustrates this behaviour with an example: for the class c_1 of word w_1, tag t_{12} is selected which is the most likely tag in an initial position. For word w_2, the tag t_{22}, the most likely tag, is selected given class c_2 and the previously selected tag t_{12}, etc.

A transducer encoding this behaviour can be generated as illustrated in Fig. 2: this shows schematically an n-type transducer in accordance with an embodiment of the invention. In this example there is a set of three classes, c_1 with the two tags t_{11} and t_{12}, c_2 with the three tags t_{21}, t_{22} and t_{23}, and c_3 with one tag t_{31}. Different classes may contain the same tag, e.g. t_{12} and t_{23} may refer to the same tag (e.g. [NOUN]). Figure 3 illustrates the procedure of building an n-type transducer in accordance with an embodiment of the invention.

Starting with the set of tags and the set of classes (generally designated 2), for every possible pair of a class and a tag (e.g. $c_1{:}t_{12}$ or [ADJ,NOUN]:NOUN) a unique state is created (step s1) and labelled with this same pair. This set of states will allow to map any class to anyone of its tags. An initial state which does not correspond with any pair, is also created in step s1. All states are final, marked by double circles. This produces a set of states labelled with class-tag pairs and 1 initial state (generally designated 4).

For every state, as many outgoing arcs are created (step s2) as there are classes (three in Fig. 2). Each such arc for a particular class points to the most probable pair of this same class. The set of outgoing arcs of one state will allow to decide on the following tag, based on the following class and the current state's tag. If the arc comes from the initial state, the most probable pair of a class and a tag (destination state) is estimated by the initial and class probability of the tag:

$$\arg\max_k \; p_1(c_i, t_{ik}) = \pi(t_{ik}) \cdot b(c_i | t_{ik})$$

If the arc comes from a state other than the initial state, the most probable pair is estimated by the transition and class probability of the tag:

$$\arg\max_k \; p_2(c_i, t_{ik}) = a(t_{ik} | t_{previous}) \cdot b(c_i | t_{ik})$$

In the example (Fig. 2), $c_1{:}t_{12}$ is the most likely pair of class c_1, and $c_2{:}t_{23}$ the most likely pair of class c_2 when coming from the initial state, and $c_2{:}t_{21}$ the most likely pair of class c_2 when coming from the state of $c_3{:}t_{31}$.

Every arc is labelled with the same symbol pair as its destination state, with the class symbol in the upper language and the tag symbol in the lower language. E.g. every arc leading to the state of $c_1{:}t_{12}$ is labelled with $c_1{:}t_{12}$. The result is the non-minimal and non-deterministic FST 6.

Finally, all state labels can be deleted since the behaviour described above is encoded in the arc labels and the network structure. The network can be minimised and determinised (step s3) to produce the n-type FST 8.

The above mathematical model is referred to as an *n1-type model*, the resulting finite-state transducer an *n1-type transducer* and the whole algorithm leading from the HMM to this transducer, an *n1-type approximation* of a first order HMM.

Adapted to a second order HMM, this algorithm would give an *n2-type approximation*.

Every *n-type* transducer is sequential, i.e. deterministic on the input side. In every state there is exactly one outgoing arc for every class symbol. An n-type transducer tags any corpus deterministically.

D. s-Type Transducers

This section presents a method, according to another embodiment of the invention, that approximates a first order Hidden Markov Model (HMM) by a finite state transducer (FST), referred to herein as an s-type approximation.

D.1 Mathematical Background

To tag a sentence i.e. to map its class sequence to the most probable tag sequence, one can split the class sequence at the unambiguous classes (containing one tag only) into subsequences, then tag them separately and concatenate them again. The result is equivalent to the one obtained by tagging the sentence as a whole.

Figure 4 is a diagram illustrating class subsequences of a sentence. Two types of subsequences of classes are distinguished: initial and middle ones. The final subsequence of a sentence is equivalent to a middle one, if it is assumed that the sentence end symbol (. or ! or ?) always corresponds to an unambiguous class c_U.

An initial subsequence C_i starts with the sentence initial position and has any number (incl. zero) of ambiguous classes and ends with the first unambiguous class c_u of the sentence. It can be described by the regular expression:

$C_i = c_a{*}\ c_u$

Given an initial class subsequence C_i of length r, its joint probability together with a corresponding initial tag subsequence T_i can be estimated by:

$$p(C_i,T_i) = \pi(t_1)\cdot b(c_1|t_1)\cdot\left[\prod_{j=2}^{r-1} a(t_j|t_{j-1})\cdot b(c_j|t_j)\right]\cdot a(t_r|t_{r-1})$$

A middle subsequence C_m starts immediately after an unambiguous class c_u, has any number (incl. zero) of ambiguous classes c_a and ends with the following unambiguous class c_u. It can be described by the regular expression:

$C_m = c_a{*}\ c_u$

To estimate the probability of middle tag sequences T_m correctly, we have to include the immediately preceding unambiguous class c_u^e actually belonging to the preceding subsequence C_i or C_m. Thus we obtain an extended middle subsequence:

$C_m^e = c_u^e\ c_a{*}\ c_u$

The joint probability of an extended middle class subsequence C_m^e of length s together with a corresponding tag subsequence T_m^e can be estimated by

$$p(C_m^e,T_m^e) = \left[\prod_{j=1}^{s-1} a(t_j|t_{j-1})\cdot b(c_j|t_j)\right]\cdot a(t_s|t_{s-1})$$

D.2 Building an s-type transducer

To build an s-type transducer, we generate a large number of initial class sequences C_i and extended middle class sequences C_m^e as described in D.3 below, and disambiguate (i.e. tag) each of them based on a first order HMM using the Viterbi algorithm for efficiency (Viterbi, 1967; Rabiner, 1990).

Every class subsequence gets linked to its most probable tag subsequence by means of the cross product operation:

$$S_i = C_i \text{ .x. } T_i = c_1\!:t_1 \quad c_2\!:t_2 \ \ldots\ldots \quad c_n\!:t_n$$

$$S_m^e = C_m^e \text{ .x. } T_m^e = c_1^e\!:t_1^e \quad c_2\!:t_2 \ \ldots\ldots \quad c_n\!:t_n$$

Then the union $^{U}S_i$ of all initial subsequences S_i and the union $^{U}S_m^e$ of all extended middle subsequences S_m^e is built.

In all extended middle subsequences S_m^e, like e.g.

$$S_m^e = \frac{C_m^e}{T_m^e} = \frac{\text{[DET] [ADJ,NOUN] [ADJ,NOUN] [NOUN]}}{\text{DET} \qquad \text{ADJ} \qquad \text{ADJ} \qquad \text{NOUN}}$$

the first class symbol on the upper side and the first tag symbol on the lower side will be marked as an extension that does not really belong to the middle sequence but is necessary to disambiguate it correctly. The above example becomes

$$S_m^0 = \frac{C_m^0}{T_m^0} = \frac{\text{0.[DET] [ADJ,NOUN] [ADJ,NOUN] [NOUN]}}{\text{0.DET} \qquad \text{ADJ} \qquad \text{ADJ} \qquad \text{NOUN}}$$

Now it is possible to formulate a preliminary sentence model containing one initial subsequence followed by any number (incl. zero) of extended middle subsequences:

$$^{U}S^0 = {}^{U}S_i \quad {}^{U}S_m^0{}^{*}$$

where all middle subsequences S_m^0 are still marked and extended in the sense that all occurrences of all unambiguous classes are mentioned twice: Once at the end of every sequence (unmarked) and also at the beginning of every following sequence (marked).

To ensure a correct concatenation of initial and middle subsequences a concatenation constraint for classes is formulated:

$$R_c = \bigcap_j \ [\sim \$[\backslash c_u \quad c_u^0]]_j$$

stating that every middle subsequence must begin with the same marked unambiguous class c_u^0 (e.g. 0.[DET]) that occurs unmarked as c_u (e.g. [DET]) at the end of the preceding subsequence since both symbols refer to the same occurrence of this unambiguous class.

Having ensured correct concatenation by composition of the preliminary sentence model ${}^{U}S^{0}$ with the concatenation constraint R_c, all marked classes on the upper side and all marked tags on the lower side of the relation are deleted.

The above mathematical model is referred to as an *s-type model*, the corresponding finite-state transducer an *s-type transducer* and the whole algorithm leading from the HMM to the transducer, an *s-type approximation* of an HMM.

An s-type transducer tags any corpus that does not contain unknown subsequences exactly the same way, i.e. with the same errors, as the corresponding HMM tagger does. An s-type transducer is, however, incomplete because it encodes only a limited set of subsequences of ambiguity classes. Therefore an s-type transducer cannot tag sentences with one or more subsequences that are not in this set. An s-type transducer can be completed as described in D.4 below.

D.3 Generation of ambiguity class subsequences

This section describes three ways to obtain class subsequences that are needed to build an s-type transducer. Figure 5 is a schematic diagram of the steps in the procedure of building an s-type transducer, in accordance with an alternative embodiment of the invention.

(a) Extraction from a corpus

Based on a lexicon and a guesser an untagged training corpus (10) is annotated with class labels, exactly the same way as is done later for the purpose of tagging.

From every sentence we extract (step s4) the initial class subsequence C_i that ends with the first unambiguous class c_u, and all extended middle subsequences C_m^0 ranging from any unambiguous class c_u in the sentence to the following unambiguous class. This generates the incomplete set of class subsequences (s-type), designated 12.

(b) Generation of all possible subsequences

Here, the set of tags and set of classes (generally designated 2) is used as the starting point. The set of all classes c is split into the subset of unambiguous classes c_u and the subset of ambiguous classes c_a. Then all possible initial and extended middle class subsequences, C_i and C_m^e up to a defined length are generated (step s5).

As in D.3(a), an incomplete set of class sequences (s-type) 12 is obtained, and the HMM can be used (step s7) to obtain from this an incomplete set of class and tag sequences (s-type) 14. This set 14 can be used with finite state calculus to build (step s8) an s-type transducer 16;

and this incomplete s-type FST in turn is used to derive the incomplete set 14, through extraction (step s10) of all class sequences.

(c) Extraction from a transducer

If an n-type transducer *N* approximating an HMM is already available, initial and extended middle class and tag subsequences, S_i and S_m^e can be extracted (step s6) using finite state calculus, to produce a complete set 17 of class and tag sequences (n-type).

D.4 Completion of s-type transducers

s-Type transducers with class subsequences that have been generated as described in D.3(a) or D.3(b) above, are in general not complete on the input side. They do not accept all possible ambiguity class sequences. This, however, is necessary for a transducer used to disambiguate class sequences of any corpus since a new corpus can contain sequences not encountered in the training corpus.

An incomplete s-type transducer *S* can be completed (step s12) with subsequences from an auxiliary complete n-type transducer *N* as follows:

First, the unions of initial and extended middle subsequences, ${}^{U}_{s}S_i$ and ${}^{U}_{s}S_m^e$ are extracted from the primary s-type transducer *S*, and the unions ${}^{U}_{n}S_i$ and ${}^{U}_{n}S_m^e$ are extracted from the auxiliary n-type transducer *N*, as described in section D.3(c) above.

Then a joint union ${}^{U}S_i$ of initial subsequences is made:

$${}^{U}S_i = {}^{U}_{s}S_i \mid [\, [\, {}^{U}_{n}S_i.u - {}^{U}_{s}S_i.u \,] .o.\, {}^{U}_{n}S_i \,]$$

and a joint union ${}^{U}S_m^e$ of extended middle subsequences:

$${}^{U}S_m^e = {}^{U}_{s}S_m^e \mid [\, [\, {}^{U}_{n}S_m^e.u - {}^{U}_{s}S_m^e.u \,] .o.\, {}^{U}_{n}S_m^e \,]$$

In both cases all subsequences from the principal model *S* are unioned with all those subsequences from the auxiliary model *N* that are not in *S*.

Finally, the complete *s+n-type* transducer 18 is generated (step s14) from the joint unions of subsequences ${}^{U}S_i$ and ${}^{U}S_m^e$, as described in section D.2 above.

A transducer completed in this way disambiguates all subsequences known to the principal incomplete s-type model exactly as the underlying HMM does, and all other subsequences as the auxiliary n-type model does.

E. b-Type Transducers

This section presents a method, according to another embodiment of the invention, that approximates a first order Hidden Markov Model (HMM) by a finite-state transducer (FST), referred to herein as b-type approximation. Regular expression operators used in this section are explained in the annex.

E.1 Basic Idea

Tagging of a sentence based on a (first order) HMM includes finding the most probable tag sequence given the class sequence of the sentence.

In this approach, an ambiguity class is disambiguated with respect to a context. A context consists of a sequence of ambiguity classes limited at both ends by some selected tag. For the left context of length β we use the term *look-back*, and for the right context of length α we use the term *look-ahead*.

In Fig. 6, the tag t^2_i can be selected from the class c_i because it is between two selected tags which are t^1_{i-2} at a look-back distance of β=2 and t^2_{i+2} at a look-ahead distance of α=2. Actually, the two selected tags t^1_{i-2} and t^2_{i+2} allow not only the disambiguation of the class c_i but of all classes inbetween, i.e. c_{i-1}, c_i and c_{i+1}.

We approximate the tagging of a whole sentence by tagging sub-sequences with selected tags at both ends (Fig. 6), and then overlapping them. The most probable paths in the tag space of a sentence, i.e. valid paths according to this approach, can be found as sketched in Fig. 7. An ordered set of overlapping sequences where every sequence is shifted by one tag to the right with respect to the previous sequence, constitutes a valid path. There can be more than one valid path in the tag space of a sentence (Fig. 7). Sets of sequences that do not overlap in such a way are incompatible according to this model, and do not constitute valid paths.

E.2 b-Type Sequences

Given a length β of look-back and a length α of look-ahead, we generate for every class c_0, every look-back sequence $t_{-\beta}\ c_{-\beta+1} \ldots c_{-1}$, and every look-ahead sequence $c_1 \ldots c_{\alpha-1}\ t_{\alpha}$, a b-type sequence:

$$t_{-\beta}\ c_{-\beta+1} \ldots c_{-1}\ c_0\ c_1 \ldots c_{\alpha-1}\ t_{\alpha}$$

For example:

CONJ [DET,PRON] [ADJ,NOUN,VERB] [NOUN,VERB] VERB

Each such *original b-type sequence* (Fig. 8) is disambiguated based on a first order HMM. Here we use the Viterbi algorithm (Viterbi, 1967; Rabiner, 1990) for efficiency.

The algorithm will be explained for a first order HMM. In the case of a second order HMM, b-type sequences must begin and end with two selected tags rather than one.

For an original *b-type sequence*, the joint probability of its class sequence *C* with its tag sequence *T* (Fig. 8), can be estimated by:

$$p(C,T) = p(c_{-\beta+1} \ldots c_{\alpha-1}, t_{-\beta} \ldots t_{\alpha}) = \left[\prod_{i=-\beta+1}^{\alpha-1} a(t_i|t_{i-1})\, b(c_i|t_i) \right] \cdot a(t_\alpha|t_{\alpha-1})$$

A boundary, i.e. a sentence beginning or end, may occur position in the look-back sequence and in the look-ahead sequence. No look-back (β=0) or no look-ahead (α=0) is also allowed. The above probability estimation can then be expressed more generally (Fig. 8) as:

$$p(C,T) = p_{start} \cdot p_{middle} \cdot p_{end}$$

with p_{start} being

$$p_{start} = a(t_{-\beta+1}|t_{-\beta}) \qquad \text{for fixed tag } t_{-\beta}$$

$$p_{start} = \pi(t_{-\beta+1}) \qquad \text{for sentence beginning \#}$$

$$p_{start} = 1 \qquad \text{for } \beta = 0\text{, i.e. no look-back}$$

with p_{middle} being

$$p_{middle} = b(c_{-\beta+1}|t_{-\beta+1}) \cdot \prod_{i=-\beta+2}^{\alpha-1} a(t_i|t_{i-1})\, b(c_i|t_i) \qquad \text{for } \beta+\alpha > 0$$

$$p_{middle} = b(c_0|t_0) \qquad \text{for } \beta+\alpha = 0$$

and with p_{end} being

$$p_{end} = a(t_\alpha|t_{\alpha-1}) \qquad \text{for fixed tag } t_\alpha$$

$$p_{end} = 1 \qquad \text{for sentence end \# or } \alpha = 0\text{, i.e. no look-ahead}$$

When the most likely tag sequence is found for an original b-type sequence, the class c_0 in the middle position is associated with its most likely tag t_0. We formulate constraints for the other tags $t_{-\beta}$ and t_α and classes $c_{-\beta+1} \ldots c_{-1}$ and $c_1 \ldots c_{\alpha-1}$ of the original b-type sequence. Thus we obtain a *tagged b-type sequence*:

$$t_{-\beta}\ c_{-\beta+1} \ldots c_{-1}\ \mathbf{c_0{:}t_0}\ c_1 \ldots c_{\alpha-1}\ t_\alpha$$

stating that t_0 is the most probable tag in the class c_0 if it is preceded by $t_{-\beta}\ c_{-\beta+1} \ldots c_{-1}$ and followed by $c_1 \ldots c_{\alpha-1}\ t_\alpha$

In the example:

CONJ-B2 [DET,PRON]-B1 [ADJ,NOUN,VERB]:ADJ [NOUN,VERB]-A1 VERB-A2

ADJ is the most likely tag in the class [ADJ,NOUN,VERB] if it is preceded by the tag CONJ two positions back (B2), by the class [DET,PRON] one position back (B1), and followed by the class [NOUN,VERB] one position ahead (A1) and by the tag VERB two positions ahead (A2).

Boundaries are denoted by a particular symbol, #, and can occur anywhere in the look-back and look-ahead. For example:

#-B2 [DET,PRON]-B1 [ADJ,NOUN,VERB]:ADJ [NOUN,VERB]-A1 VERB-A2

CONJ-B2 [DET,PRON]-B1 [ADJ,NOUN,VERB]:NOUN #-A1

Note that look-back length β and look-ahead length α also include all sequences shorter than β or α, respectively, that are limited by a boundary #.

Figure 9 is a schematic diagram illustrating the steps in the procedure of building a b-type transducer, according to an embodiment of the invention.

For a given length β of look-back and a length α of look-ahead, and using the set of tags and the set of classes, every possible *original b-type sequence* is generated (step s90). Then, these are disambiguated statistically, and encoded it in a *tagged b-type sequence* B_i as described above (Viterbi; step s92). All sequences B_i are then unioned and a preliminary tagger model B' generated (step s94):

$$B' = \left[\bigcup_i B_i \right]^*$$

where all sequences B_i can occur in any order and number (including zero times) because no constraints have yet been applied.

E.3 Concatenation Constraints

To ensure a correct concatenation of sequences, it is necessary to make sure that every sequence B_i is preceded and followed by other B_i according to what is encoded in the look-back and look-ahead which were explained above.

Constraints are created for preceding and following tags, classes and sentence boundaries. For the look-back, a particular tag t_i or class c_j is requested for a particular distance of $\delta \leq -1$, by:

$$R^{\delta}(t_i) = \sim[\ \sim[?^*\ t_i\ [\backslash^{\cup}t]^*\ \ [^{\cup}t\ \ [\backslash^{\cup}t]^*]\wedge(-\delta-1)\ \ t_i^{B(-\delta)}\ ?^*]$$

$$R^{\delta}(c_j) = \sim[\ \sim[?^*\ c_j\ [\backslash^{\cup}c]^*\ \ [^{\cup}c\ \ [\backslash^{\cup}c]^*]\wedge(-\delta-1)\ \ c_j^{B(-\delta)}\ ?^*]$$

for δ≤ -1

with $^{\cup}t$ and $^{\cup}c$ being the union of all tags and all classes respectively. A sentence beginning, #, is requested for a particular look-back distance of δ≤−1, by:

$$R^{\delta}(\#) = \sim[\ \sim[\ [\backslash^{\cup}t]^*\ \ [^{\cup}t\ \ [\backslash^{\cup}t]^*]\wedge(-\delta-1)\ \ \#^{B(-\delta)}\ ?^*]$$

for δ≤−1

In the case of look-ahead, for a particular distance of δ≥1, a particular tag t_i or class c_j or a sentence end # is required in a similar way, by:

$$R^{\delta}(t_i) = \sim[?^*\, t_i^{A\delta}\ \ \sim[[\backslash^{\cup}t]^*\ \ [^{\cup}t\ \ [\backslash^{\cup}t]^*]\wedge(\delta-1)\ \ t_i\ ?^*]]$$

$$R^{\delta}(c_j) = \sim[?^*\, c_j^{A\delta}\ \ \sim[[\backslash^{\cup}c]^*\ \ [^{\cup}c\ \ [\backslash^{\cup}c]^*]\wedge(\delta-1)\ \ c_j\ ?^*]]$$

$$R^{\delta}(\#) = \sim[?^*\, \#^{A\delta}\ \ \sim[[\backslash^{\cup}t]^*\ \ [^{\cup}t\ \ [\backslash^{\cup}t]^*]\wedge(\delta-1)\,]]$$

for δ≥1

We create the intersection R_t of all tag constraints $R^{\delta}(t_i)$, the intersection R_c of all class constraints $R^{\delta}(c_j)$, and the intersection $R_{\#}$ of all sentence boundary constraints $R^{\delta}(\#)$:

$$R_t = \bigcap_{\delta,i} R^{\delta}(t_i)$$

$$R_c = \bigcap_{\delta,j} R^{\delta}(c_j)$$

$$R_{\#} = \bigcap_{\delta} R^{\delta}(\#)$$

All constraints (for tags, classes and sentence boundaries) are enforced (step s96 in Fig. 9) by composition with the preliminary tagger model B'. The class constraint R_c must be composed on the upper side of B' which is the side of the classes, and both the tag constraint R_t and the boundary constraint $R_{\#}$ must be composed on the lower side of B' which is the side of the tags:

$$B'' = R_c \ .o.\ B'\ .o.\ R_t\ .o.\ R_{\#}$$

Having ensured correct concatenation, all symbols that have served to constrain tags, classes or boundaries are deleted (from *B'*)(step s98). Finally, the FST is determinized and minimized.

The above-mentioned mathematical model is referred to as a *b-type model*, the corresponding FST as *a b-type transducer*, and the whole algorithm leading from the HMM to the transducer, as a *b-type approximation* of an HMM.

E.4 Properties of b-Type Transducers

There are two groups of b-type transducers with different properties: FSTs without look-back or without look-ahead and FSTs with both look-back and look-ahead. Both accept any sequence of ambiguity classes.

b-Type FSTs without look-back or without look-ahead are always sequential. They map a class sequence that corresponds to the word sequence of a sentence, always to exactly one tag sequence. Their tagging accuracy and similarity with the underlying HMM increases with growing look-back or look-ahead. A b-type FST with look-back $\beta=1$ and without look-ahead ($\alpha=0$) is equivalent to an n1-type FST (section C).

b-Type transducers with both look-back and look-ahead are in general not sequential. For a class sequence corresponding to the word sequence of a sentence, they deliver a set of alternative tag sequences, which means that the tagging results are ambiguous. This set is never empty, and the most probable tag sequence according to the underlying HMM is always in this set. The longer the look-back distance β and the look-ahead distance α are, the larger the FST and the smaller the set of alternative tagging results. For sufficiently large look-back plus look-ahead, this set may contain always only one tagging result. In this case the b-type FST is equivalent to the underlying HMM. For reasons of size however, this FST is only computable for HMMs with small tag sets.

F. An Implemented Finite State Tagger

The implemented tagger requires three transducers which represent a lexicon, a guesser and an approximation of an HMM. Figure 10 is an illustration of the tagging of a sentence using either the n-type transducer of Fig. 3, the s-type transducer formed in Fig. 5 or the b-type transducer formed in Fig. 9. Figure 11 is a schematic flow chart of the steps involved in the procedure, in accordance with an embodiment of the invention, of tagging a text corpus 20 with a finite state tagger using an HMM transducer.

Every word 21 of an input sentence is read (step s71) from the corpus and is initially looked for in the lexicon; and if this fails, the search continues in the guesser (step s72), resulting in a word labelled with a class (22). As soon as an input token gets labelled with the sentence end class (e.g. [SENT] in Fig. 10), the tagger stops (step s73) reading words from the input. At that point the tagger has read and stored the words of a whole sentence (Fig. 6, col. 1) and generated the corresponding sequence of classes 23 (see Fig. 10, col. 2).

The class sequence 23 is now deterministically mapped (step s74) to a tag sequence 24 (see Fig. 10, col. 3) by means of the HMM transducer.

The tagger outputs (step s75) the stored word and tag sequence 24 of the sentence and continues (step s76) the same way with the remaining sentences of the corpus, stopping (step s77) when the end of the corpus is reached. The end result is the tagged text corpus 25.

G. Tests and Results

Table 1 compares an n-type and an s-type transducer with the underlying HMM on an English test case. As expected, the transducers perform tagging faster than the HMM.

	tagging accuracy in %	tagging speed on different computers in words/sec		transducer size	
		on ULTRA2	on SPARC20	num. states	num. arcs
HMM	96.77	4 590	1 564	none	none
s-type transducer	95.05	12 038	5 453	4 709	976 785
n-type transducer	94.19	17 244	8 180	71	21 087

Language:	English
HMM training corpus:	19 944 words
Test corpus:	19 934 words
Tag set:	74 tags 297 classes
s-Type transducer	with subsequences from a training corpus of 100,000 words completed with subsequences from an n-type transducer

Table 1: Accuracy, speed, size and creation time of HMM transducers

Since both transducers are approximations of HMMs, they show a slightly lower tagging accuracy than the HMMs. However, improvement in accuracy can be expected since these transducers can be composed with transducers encoding correction rules for frequent errors, as described above in the introductory part of this disclosure.

The s-type transducer is more accurate in tagging but also larger and slower than the n-type transducer.

Table 2 compares the tagging accuracy of n-type and s-type transducers and the underlying HMM for different languages.

	Tagging accuracy in %					
	English	Dutch	French	German	Portug.	Spanish
HMM	96.77	94.76	98.65	97.62	97.12	97.60
s-type transducer	95.05	92.36	98.37	95.81	96.56	96.87
n-type transducer	94.19	91.58	98.18	94.49	96.19	96.46

s-Type transducer	with subsequences from a training corpus of 100,000 words completed with subsequences from n-type transducer

Table 2: Accuracy of HMM transducers for different languages

For the test of b-type transducers we used an English corpus, lexicon and guesser, that originally were annotated with 74 different tags. We automatically recoded the tags in order to reduce their number, i.e. in some cases more than one of the original tags were recoded into one and the same new tag. We applied different recodings, thus obtaining English corpora, lexicons and guessers with reduced tag sets of 45, 36, 27, 18 and 9 tags respectively.

Transducer or HMM	Accuracy	Tagging speed		Transducer size		Creation
	test corp.	in words/second		num.of	num.of	time
	in %	on Ultra2	onSparc20	states	arcs	on Ultra2
HMM	97.07	3 358	1 363	—	—	—
b-FST (β=0,α=0)	94.47	25 521	11 815	1	119	3 sec
b-FST (β=1,α=0)	95.60	25 521	12 038	28	3 332	4 sec
b-FST (β=2,α=0)	97.71	25 521	11 193	1 642	195 398	32 min
b-FST (β=0,α=1)	95.26	17 244	9 969	137	14 074	5 sec
b-FST (β=0,α=2)	95.37	19 939	9 969	3 685	280 545	3 min
b-FST (β=1,α=1)	*96.78	16 790	8 986	1 748	192 275	19 sec
b-FST (β=2,α=1)	*97.06	22 787	11 000	19 878	1 911 277	61 min

Language	English
Corpora	19 944 words for HMM training, 19 934 words for test
Tag set	27 tags, 119 classes
*	Multiple, i.e. ambiguous tagging results: Only first result retained
Types of FST (Finite State Transducers)	
b (β=2,α=1)	b-type transducer with look-back of 2 and look-ahead of 1
Computers	
Ultra2	1 CPU, 512 MBytes physical RAM, 1.4 GBytes virtual RAM
Sparc20	1 CPU, 192 MBytes physical RAM, 827 MBytes virtual RAM

Table 3: Accuracy, tagging speed and size of some HMM transducers

Table 3 compares b-type transducers with different length of look-back and look-ahead for a tag set of 27 tags. The highest accuracy (97.06 %) could be obtained with an b-type FST with β=2 and α=1. This b-type FST produced in some cases ambiguous tagging results. Then only the first result found was retained.

Transducer or HMM	Tagging accuracy with tag sets of different sizes (in %)					
	74 tags 297 cls.	45 tags 214 cls.	36 tags 181 cls.	27 tags 119 cls.	18 tags 97 cls.	9 tags 67 cls.
HMM	96.78	96.92	97.35	97.07	96.73	95.76
b-FST (β=0,α=0)	83.53	83.71	87.21	94.47	94.24	93.86
b-FST (β=1,α=0)	94.19	94.09	95.16	95.60	95.17	94.14
b-FST (β=2,α=0)	—	94.28	95.32	95.71	95.31	94.22
b-FST (β=0,α=1)	92.79	92.47	93.69	95.26	95.19	94.64
b-FST (β=0,α=2)	93.46	92.77	93.92	95.37	95.30	94.80
b-FST (β=1,α=1)	*94.94	*95.14	*95.78	*96.78	*96.59	*95.36
b-FST (β=2,α=1)	—	—	*97.34	*97.06	*96.73	*95.73
b-FST (β=3,α=1)	—	—	—	—	—	95.76

Language English
Corpora 19 944 words for HMM training, 19 934 words for test
Types of FST (Finite State Transducers) see Table 3
* Multiple, i.e. ambiguous tagging results: Only first result retained
— Transducer could not be computed for reasons of size.

Table 4: Tagging accuracy with tags sets of different sizes

Table 4 shows the tagging accuracy of different b-type transducers with tag sets of different sizes. To get results that are almost equivalent to those of an HMM, a b-type FST needs at least a look-back of β=2 and a look-ahead of α=1 or vice versa. For reasons of size, this kind of FST could only be computed for the tag sets with 36 tags or less. A b-type FST with β=3 and α=1 could only be computed for the tag set with 9 tags. This FST gave exactly the same tagging results as the underlying HMM.

H. References

Ait-Mokhtar, Salah and Chanod, Jean-Pierre (1997). Incremental Finite-State Parsing. In: *Proceedings of the 5^{th} Conference of Applied Natural Language Processing*. ACL, pp. 72-79. Washington, DC, USA.

Bahl, Lalit R. and Mercer, Robert L. (1976). Part of Speech Assignment by a Statistical Decision Algorithm. In: *IEEE international Symposium on Information Theory*. pp. 88-89. Ronneby.

Brill, Eric (1992). A Simple Rule-Based Part-of-Speech Tagger. In: *Proceedings of the 3^{rd} conference on Applied Natural Language Processing*, pp. 152-155. Trento, Italy.

Chanod, Jean-Pierre and Tapanainen, Pasi (1995). Tagging French - Comparing a Statistical and a Constraint Based Method. In: *Proceedings of the 7th conference of the EACL*. pp. 149-156. ACL. Dublin, Ireland.

Church, Kenneth W. (1988). A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In: *Proceedings of the 2nd Conference on Applied Natural Language Processing.* ACL, pp. 136-143.

Kaplan, Ronald M. and Kay, Martin (1994). Regular Models of Phonological Rule Systems. In: *Computational Linguistics.* 20:3, pp. 331-378.

Rabiner, Lawrence R. (1990). *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* In: *Readings in Speech Recognition* (eds. A. Waibel, K.F. Lee). Morgan Kaufmann Publishers, Inc. San Mateo, CA., USA.

Viterbi, A.J. (1967). Error Bounds for Convolutional Codes and an Asymptotical Optimal Decoding Algorithm. In: *Proceedings of IEEE*, vol. 61, pp. 268-278.

Annex: Regular Expression Operators of the XEROX Finite State Calculus

Below, **a** and **b** designate symbols, **A** and **B** designate languages, and **R** and **Q** designate relations between two languages. More details on the following operators and pointers to finite-state literature can be found in `http://www.rxrc.xerox.com/research/mltt/fst`

`$A` **Contains.** Set of strings containing at least one occurrence of a string from **A** as a substring.

`~A` **Complement** (negation). All strings except those from **A.**

`\a` **Term complement.** Any symbol other than **a.**

`A*` **Kleene star.** Zero or more times **A** concatenated with itself.

`A^n` **A n times.** Language **A** concatenated **n** times with itself.

`A+` **Kleene plus.** One or more times **A** concatenated with itself.

`a -> b` **Replace.** Relation where every **a** on the upper side gets mapped to a **b** on the lower side.

`a <- b` **Inverse replace.** Relation where every **b** on the lower side gets mapped to an **a** on the upper side.

`a:b` **Symbol pair** with **a** on the upper and **b** on the lower side.

`R.u` **Upper language** of the relation **R.**

`R.l` **Lower language** of the relation **R.**

`R.i` **Inverse** relation where the upper and the lower language are exchanged with respect to **R.**

`A B` **Concatenation** of all strings of **A** with all strings of **B.**

`A | B` **Union** of the languages **A** and **B.**

`A & B` **Intersection** of the languages **A** and **B.**

`A - B` **Relative complement** (minus). All strings of the language **A** that are not in **B.**

`A .x. B` **Cross Product** (Cartesian product) of the languages **A** and **B.**

`R .o. Q` **Composition** of the relations **R** and **Q.**

`0 or [ ]` **Empty string** (epsilon).

`?` **Any symbol** in the known alphabet and its extensions.

CLAIMS:

1. A method carried out in a text processing system, for generating a text tagging finite state transducer (FST) for a predetermined language, the finite state transducer encoding along a plurality of arcs sets of ordered pairs of upper and lower labels, comprising:

   (a) providing a set of tags, the or each tag being a part-of-speech designator for the language, and a set of classes, the or each class being ambiguity classes, defining groups of possible tags, for words in the language,

   (b) providing a plurality of states, including an initial state and, for each class-tag pair, a further state labelled as the respective class-tag pair, and

   (c) for each state, for each class, creating an arc between the state and a destination state labelled with a class-tag pair, the class tag pair being the most probable for the class, the arc so added having the class as its upper label and a tag of that class as its lower label.

2. The method of claim 1, wherein (c) comprises, where the state is the initial state, (c1) determining the most probable class tag pair using

$$\arg\max_k p_1(c_i, t_{ik}) = \pi(t_{ik}) \cdot b(c_i | t_{ik})$$

where π is initial probability, and b is class probability.

3. The method of claim 1 or 2, wherein (c) comprises, where the state is not the initial state, (c1') determining the most probable class tag pair using

$$\arg\max_k p_2(c_i, t_{ik}) = a(t_{ik} | t_{previous}) \cdot b(c_i | t_{ik})$$

where a is transition probability, and b is class probability.

4. The method of claim 1, 2 or 3, further comprising minimising and determinising the FST generated in steps (a)- (c).

5. A method carried out in a text processing system, for generating a text tagging finite state transducer (FST) for a predetermined language, the finite state transducer encoding along a plurality of arcs sets of ordered pairs of upper and lower labels, comprising:

   (a) providing an incomplete set of class sequences, the class sequences being sequences of classes, the or each class being ambiguity classes, defining groups of possible tags, for words in the language,

   (b) tagging the class sequences using a Hidden Markov Model, to produce an incomplete set of class and tag sequences of a first type,

   (c) providing a complete set of class and tag sequences of a second type,

(d) combining the set of class and tag sequences of a first type with the set of class and tag sequences of a second type, and

(e) building a FST using the combination obtained in step (d).

6. The method of claims 5, wherein step (a) comprises:

(a1) providing a set of tags and a set of classes, and

(a2) creating from said set of tags and set of classes, all possible class sequences up to a predetermined length.

7. The method of claim 5, wherein step (a) comprises:

(a1') providing an untagged text corpus, and

(a2') extracting all class sequences from the untagged text corpus.

8. The method of claim 5, 6 or 7, wherein step © comprises:

(c1) providing a FST generated according to the method of any of claims 1 to 4,

(c2) extracting all class sequences from the FST provided in step (c1).

9. A method carried out in a text processing system, for generating a text tagging finite-state transducer (FST) for a predetermined language, the FST encoding along a plurality of arcs sets of ordered pairs of upper and lower labels, comprising:

(a) providing a set of tags, the or each tag being a part-of-speech designator for the language, and a set of classes, the or each class being an ambiguity class, defining groups of possible tags, for the language,

(b) creating from the set of tags and the set of classes a set of subsequences of ambiguity classes, said set of subsequences comprising all possible subsequences for a predetermined look-back value and look-ahead value,

(c) using a Hidden Markov Model, tagging said set of subsequences to generate a set of tagged subsequences,

(d) performing a union operation, followed by a Kleene star operation, on the set of tagged subsequences, to generate a preliminary tagger model, said preliminary tagger model being a sequence in which the tagged subsequences may appear in any order or in any number,

(e) composing a plurality of constraints with the preliminary tagger model to generate the tagging FST, said constraints including constraints for tags, classes and sentence boundaries, and deleting all symbols expressing constraints.

10. A method carried out in a text processing system, fore tagging an untagged text corpus, comprising:

(a) reading, in sequence, the or each word of the text corpus,

(b) for the or each word, looking up the word using a lexical resource, to obtain the word labelled with its class,

(c) if a sentence end token has not been read, repeating steps (a) and (b), the sentence end token defining the end of the current sentence,

(d) if a sentence end token has been read, applying the FST obtainable by the method of claims 1, 5 or 9 to the class sequence corresponding to the current sentence, to generate a tag sequence, the tag sequence comprising a sequence of tags,

(e) outputting a tagged form of the current sentence, the tagged form comprising the or each word of the current sentence and, appended thereto, a respective tag from said tag sequence, and

(f) if the end of the text corpus has not been reached, repeating steps (a)-(e).

11. A text processing system when suitably programmed for carrying out the method of any of the preceding claims, the system comprising a processor and memory, the processor being operable with said memory for executing instructions corresponding to steps of any of said methods.

12. A recordable medium having recorded thereon digital data defining instructions for execution by a processor and corresponding the steps of the methods of any of claims 1 to 10.

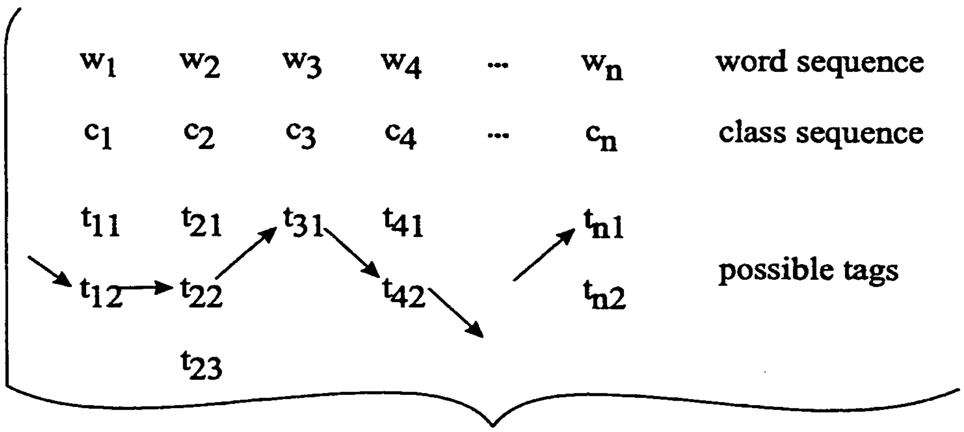

w1 w2 w3 w4 ... wn
word sequence
c1 c2 c3 c4 ... cn
class sequence
t11 t21 t31 t41 tn1
t12 t22 t42 tn2
t23
possible tags

FIG. 1

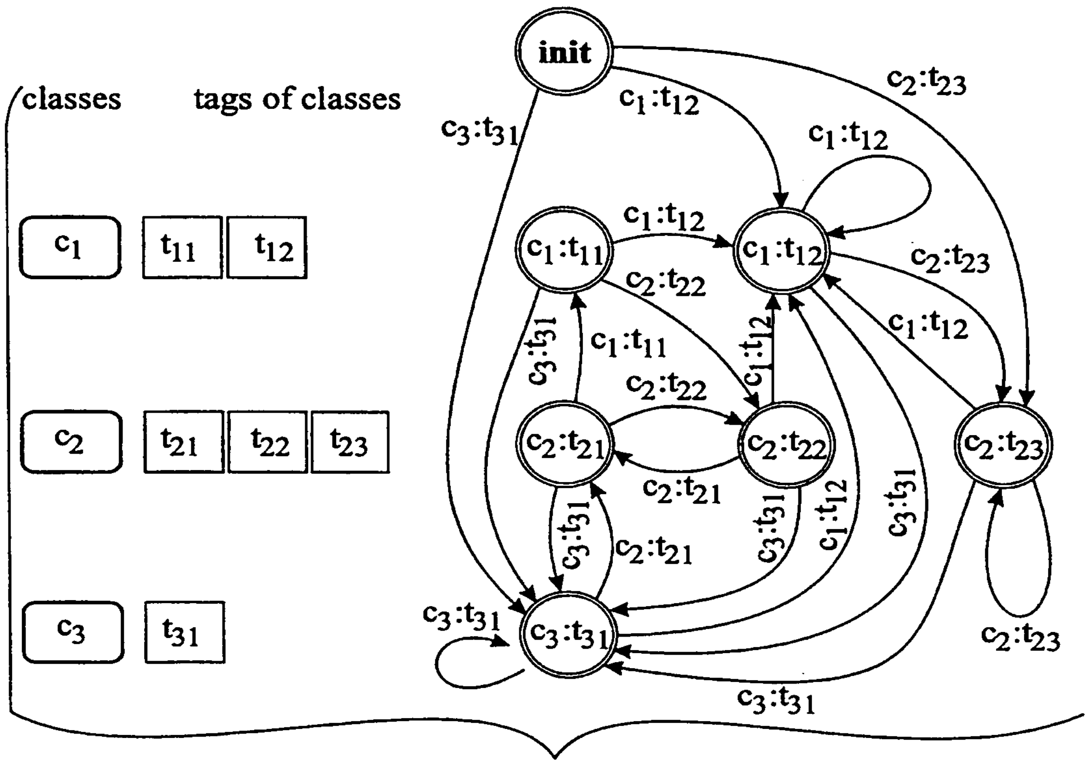

classes
tags of classes
c_1
t_{11}
t_{12}
c_2
t_{21}
t_{22}
t_{23}
c_3
t_{31}
init
$c_1:t_{11}$
$c_1:t_{12}$
$c_2:t_{21}$
$c_2:t_{22}$
$c_2:t_{23}$
$c_3:t_{31}$

FIG. 2

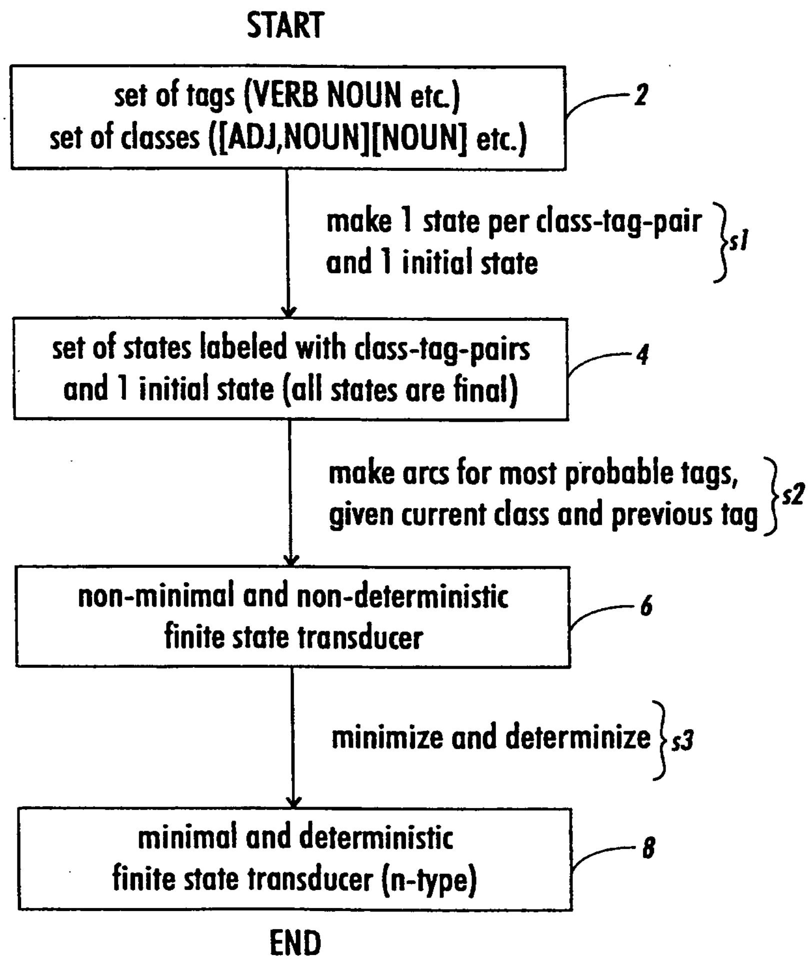

START
set of tags (VERB NOUN etc.)
set of classes ([ADJ,NOUN][NOUN] etc.)
2
make 1 state per class-tag-pair
and 1 initial state
s1
set of states labeled with class-tag-pairs
and 1 initial state (all states are final)
4
make arcs for most probable tags,
given current class and previous tag
s2
non-minimal and non-deterministic
finite state transducer
6
minimize and determinize
s3
minimal and deterministic
finite state transducer (n-type)
8
END

FIG. 3

w_1	w_2	w_3	w_4	w_5	w_6	...	w_n	word sequence
c_1	c_2	c_3	c_4	c_5	c_6	...	c_n	class sequence
c_a	c_a	c_u						initial subsequence
			c_a	c_a	c_u			middle subsequence (1st)
						...	c_u	middle subsequence (last)
		c_u^e	c_a	c_a	c_u			extended middle subsequences
					c_u^e	...		
						...	c_u	

FIG. 4

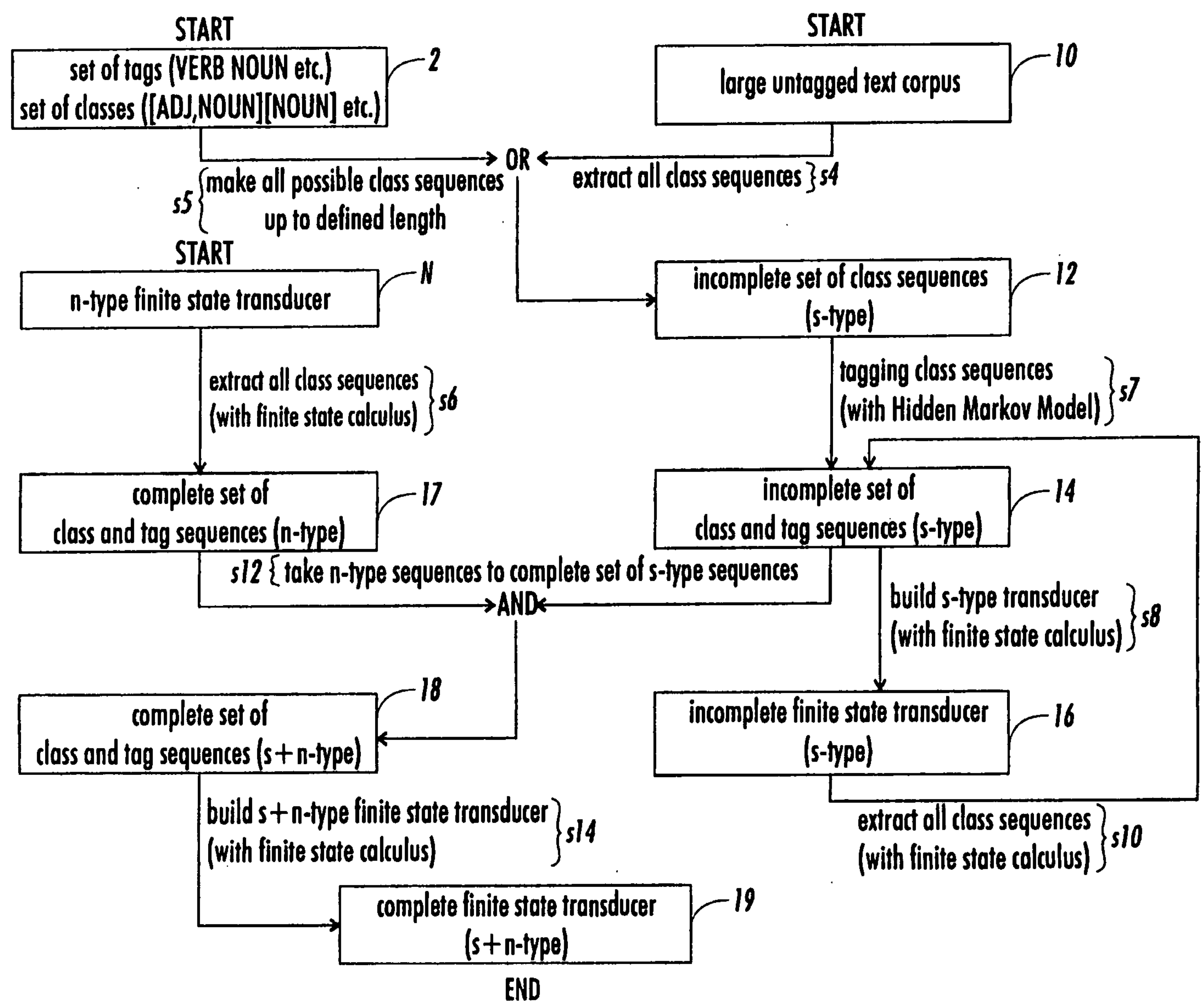

START
set of tags (VERB NOUN etc.)
set of classes ([ADJ,NOUN][NOUN] etc.)
2
START
large untagged text corpus
10
OR
s5
make all possible class sequences
up to defined length
extract all class sequences
s4
START
n-type finite state transducer
N
incomplete set of class sequences
(s-type)
12
extract all class sequences
(with finite state calculus)
s6
tagging class sequences
(with Hidden Markov Model)
s7
complete set of
class and tag sequences (n-type)
17
incomplete set of
class and tag sequences (s-type)
14
s12
take n-type sequences to complete set of s-type sequences
AND
build s-type transducer
(with finite state calculus)
s8
complete set of
class and tag sequences (s+n-type)
18
incomplete finite state transducer
(s-type)
16
build s+n-type finite state transducer
(with finite state calculus)
s14
extract all class sequences
(with finite state calculus)
s10
complete finite state transducer
(s+n-type)
19
END

FIG. 5

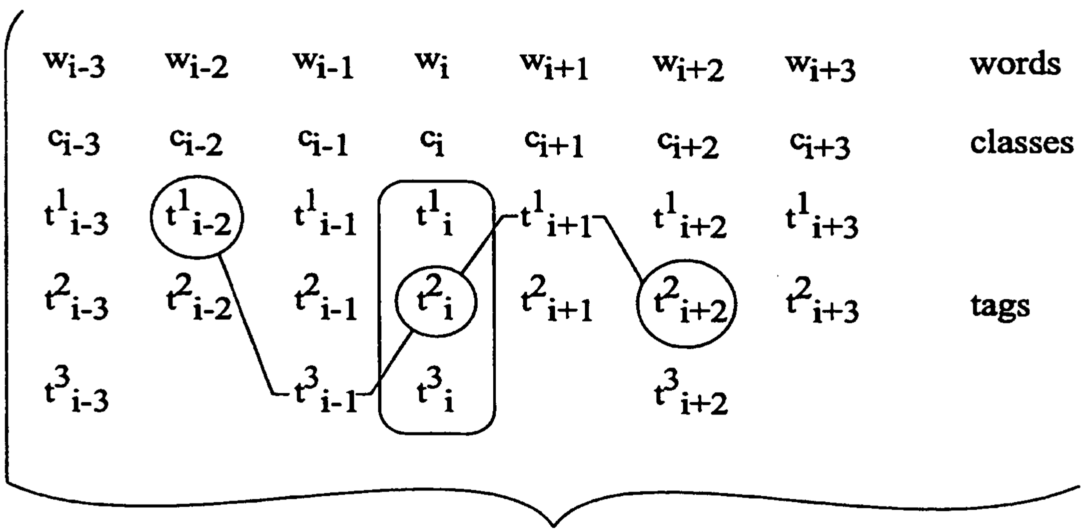

w_{i-3} w_{i-2} w_{i-1} w_i w_{i+1} w_{i+2} w_{i+3} words
c_{i-3} c_{i-2} c_{i-1} c_i c_{i+1} c_{i+2} c_{i+3} classes
t^1_{i-3} t^1_{i-2} t^1_{i-1} t^1_i t^1_{i+1} t^1_{i+2} t^1_{i+3}
t^2_{i-3} t^2_{i-2} t^2_{i-1} t^2_i t^2_{i+1} t^2_{i+2} t^2_{i+3} tags
t^3_{i-3} t^3_{i-1} t^3_i t^3_{i+2}

FIG. 6

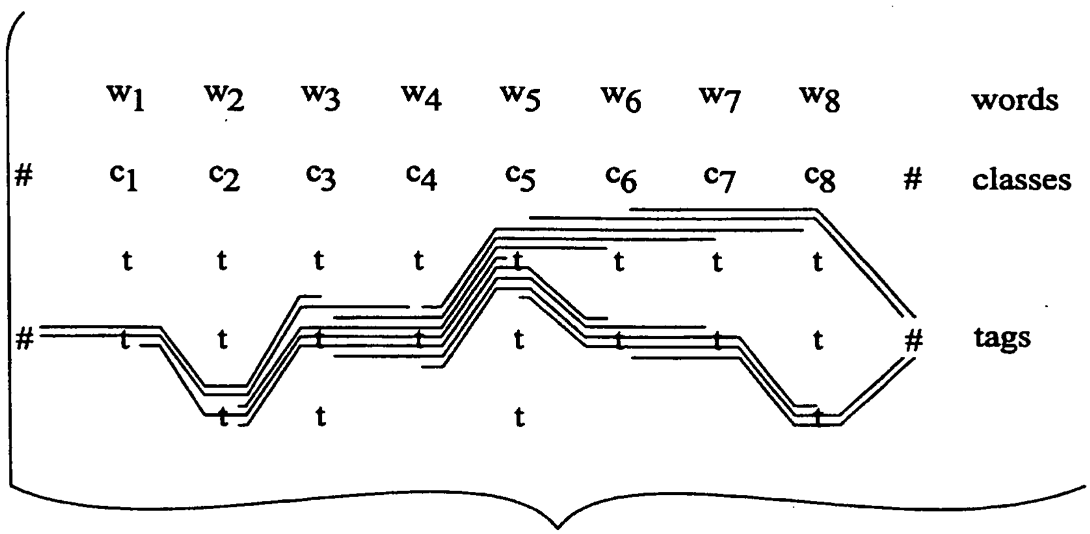

w_1 w_2 w_3 w_4 w_5 w_6 w_7 w_8
words
c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8
classes
t t t t t t t t
t t t t t t t t
tags
t t t

FIG. 7

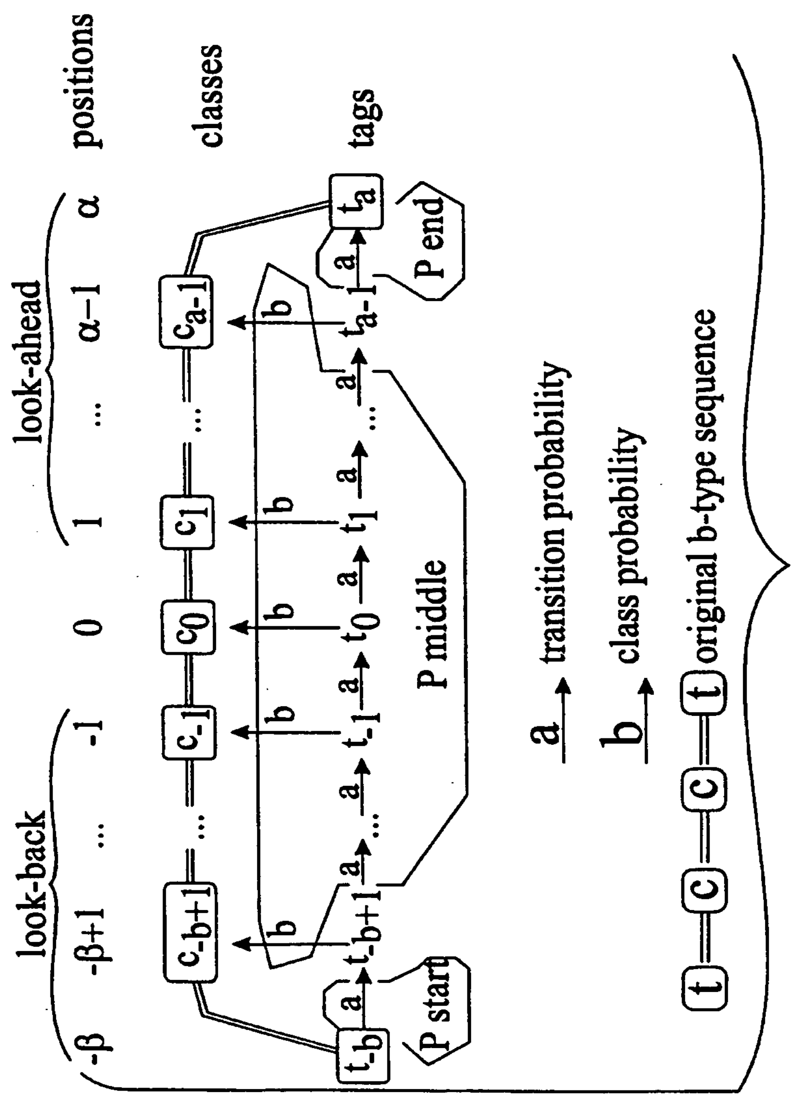

look-back
look-ahead
-β
-β+1
...
-1
0
1
...
α-1
α
positions
classes
tags
P start
P middle
P end
transition probability
class probability
original b-type sequence

FIG. 8

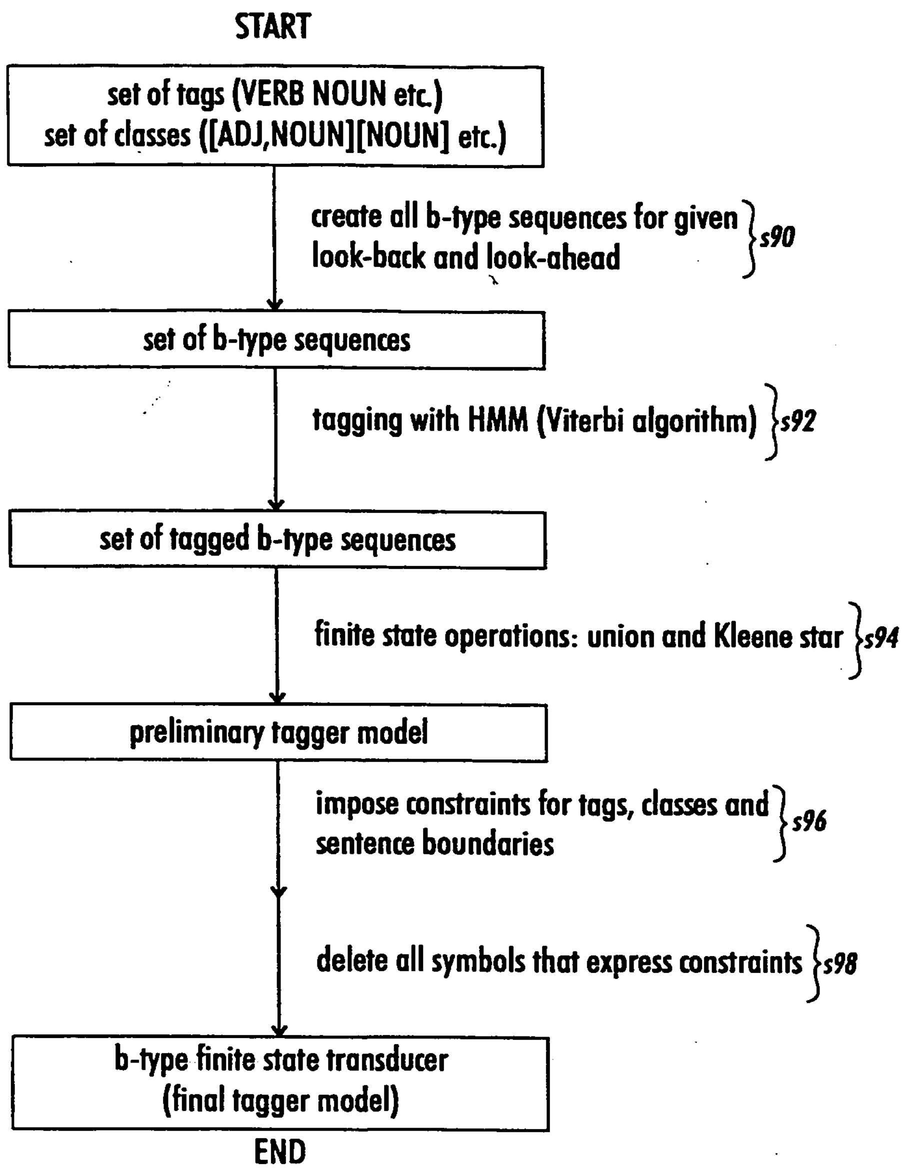

START
set of tags (VERB NOUN etc.)
set of classes ([ADJ,NOUN][NOUN] etc.)
create all b-type sequences for given look-back and look-ahead
s90
set of b-type sequences
tagging with HMM (Viterbi algorithm)
s92
set of tagged b-type sequences
finite state operations: union and Kleene star
s94
preliminary tagger model
impose constraints for tags, classes and sentence boundaries
s96
delete all symbols that express constraints
s98
b-type finite state transducer
(final tagger model)
END

FIG. 9

The	[AT]	AT
share	[NN,VB]	NN
of	[IN]	IN
the	[AT]	AT
new	[JJ,RB]	JJ
housing	[NN,VBG]	NN
market	[NN,VB]	NN
enjoyed	[VBD,VBN]	VBD
by	[IN,RB]	IN
apartments	[NNS]	NNS
has	[HVZ]	HVZ
more	[AP,RB]	RB
than	[CS,IN]	CS
tripled	[VBD,VBN]	VBD
within	[IN,RB]	IN
that	[CS,DT,WPS]	DT
span	[NN,VB,VBD]	VBD
of	[IN]	IN
time	[NN,VB]	NN
.	[SENT]	SENT

FIG. 10

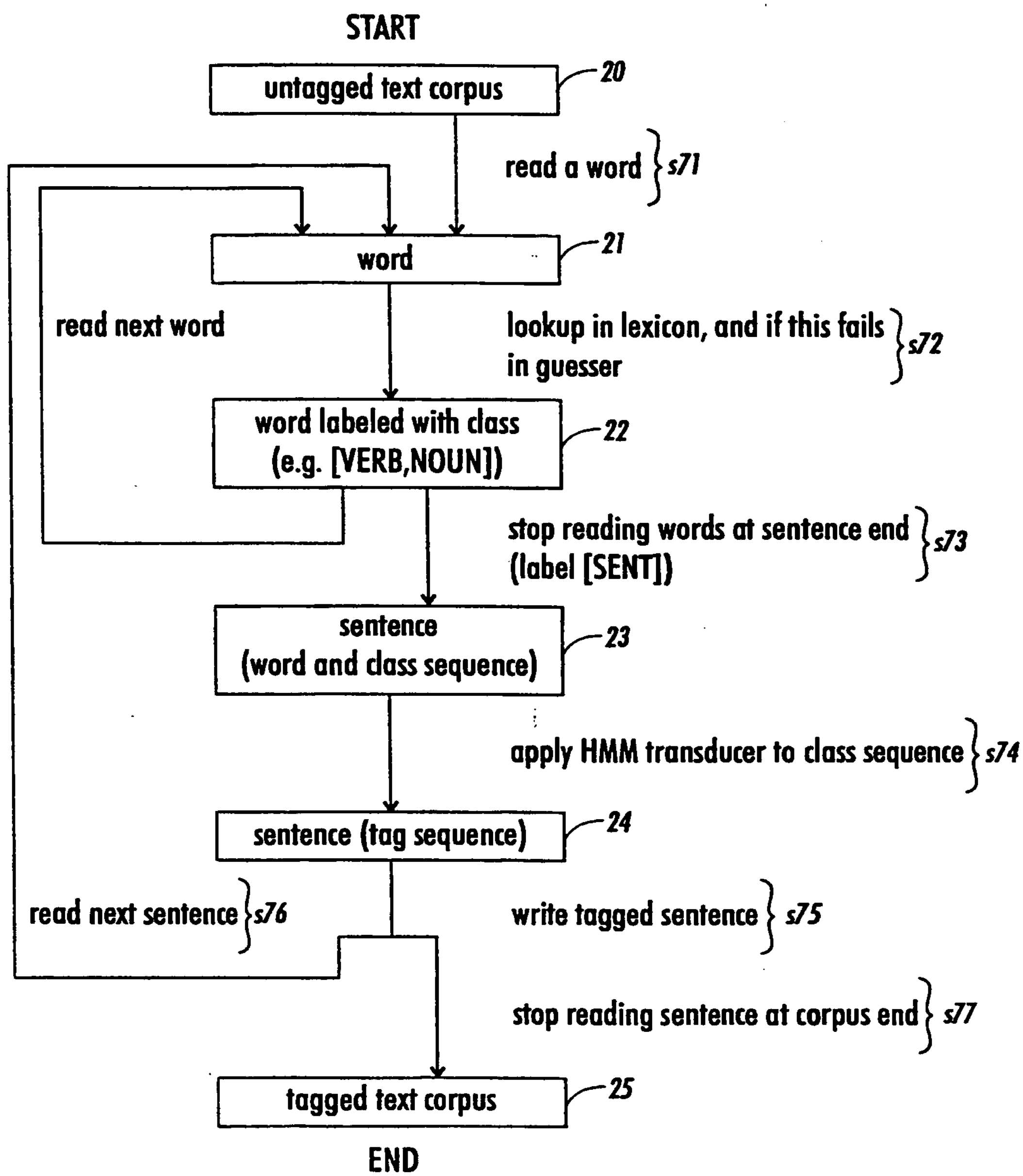

START
untagged text corpus
20
read a word
s71
word
21
read next word
lookup in lexicon, and if this fails in guesser
s72
word labeled with class (e.g. [VERB,NOUN])
22
stop reading words at sentence end (label [SENT])
s73
sentence (word and class sequence)
23
apply HMM transducer to class sequence
s74
sentence (tag sequence)
24
read next sentence
s76
write tagged sentence
s75
stop reading sentence at corpus end
s77
tagged text corpus
25
END

FIG. 11

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F17/27

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	A. KEMPE: "Finite State Transducers Approximating Hidden Markov Models." PROCEEDINGS OF THE 35TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 7 - 12 July 1997, pages 460-467, XP002085043 Madrid, Spain see the whole document --- -/--	1-8, 10-12

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 November 1998	04/12/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Pedersen, N

1

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
P,X	A. KEMPE: "Look-Back and Look-Ahead in the Conversion of Hidden Markov Models into finite state transducers." PROCEEDINGS OF THE JOINT CONFERENCE ON NEW METHODS IN LANGUAGE PROCESSING AND COMPUTATIONAL NATURAL LANGUAGE LEARNING, 15 - 17 January 1998, pages 29-37, XP002085044 Sydney, (http://www.xrce.xerox.com/publis/mltt/mltart.html) Available on the internet 10th October 1998 see the whole document	9-12
P,X	CHARNIAK E: "Statistical techniques for natural language parsing" AI MAGAZINE, WINTER 1997, AMERICAN ASSOC. ARTIFICIAL INTELLIGENCE, USA, vol. 18, no. 4, pages 33-44, XP002085045 ISSN 0738-4602 see page 34, column 1, line 3 - page 35, column 1, line 13 see page 39, column 2, line 5 - page 40, column 1, line 1 see figure 4	1-4, 10-12
X	PEREIRA F ET AL: "WEIGHTED RATIONAL TRANSDUCTION AND THEIR APPLICATION TO HUMAN LANGUAGE PROCESSING" HUMAN LANGUAGE TECHNOLOGY. PROCEEDINGS OF A WORKSHOP,1 January 1994, pages 262-267, XP000571116	1-3, 10-12
Y	see page 262, column 1, line 1 - page 263, column 1, line 34	5-9
Y	TAPANAINEN P ET AL: "Syntactic Analysis of Natural Language using Linguistic Rules and Corpus-Based Patterns" PROCEEDINGS OF THE FIFTEENTH INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS (COLING94) , vol. 1, 1994, pages 629-634, XP002085046 Kyoto, Japan see page 629, column 1, line 18 - column 1, line 45 see page 629, column 2, line 20 - line 35 see page 631, column 2, line 19 - line 59	5-9
X	US 5 610 812 A (SCHABES YVES ET AL) 11 March 1997 see column 6, line 63 - column 12, line 37 see figures 2-7	1,4, 10-12

-/--

1

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	ROCHE E ET AL: "Deterministic part-of-speech tagging with finite-state transducers" COMPUTATIONAL LINGUISTICS, JUNE 1995, USA, vol. 21, no. 2, pages 227-253, XP002085047 ISSN 0891-2017 see page 231, line 6 - page 234, line 12 see page 236, line 16 - line 26 see page 242, line 1 - page 243, line 24 ---	1,4, 10-12
A	KAPLAN R.M ET AL: "REGULAR MODELS OF PHONOLOGICAL RULE SYSTEMS" COMPUTATIONAL LINGUISTICS, vol. 20, no. 3, 1 September 1994, pages 331-378, XP000569407 see page 333, line 46 - page 338, line 36 ---	1-12
A	RABINER L R: "A tutorial on hidden Markov models and selected applications in speech recognition" PROCEEDINGS OF THE IEEE, FEB. 1989, USA, vol. 77, no. 2, pages 257-286, XP002085048 ISSN 0018-9219 see page 258, column 2, line 3 - page 259, column 1, line 26 -----	1-12

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5610812 A	11-03-1997	JP 8055122 A	27-02-1996